Recursion

Fibonacci Series

1st term → 0
2nd term → 1

0 1 1 2 3 5

fib(3) = fib(2) + fib(1)
fib(4) = fib(3) + fib(2)
fib(5) = fib(4) + fib(3)

fib(n) = fib(n-1) + fib(n-2)

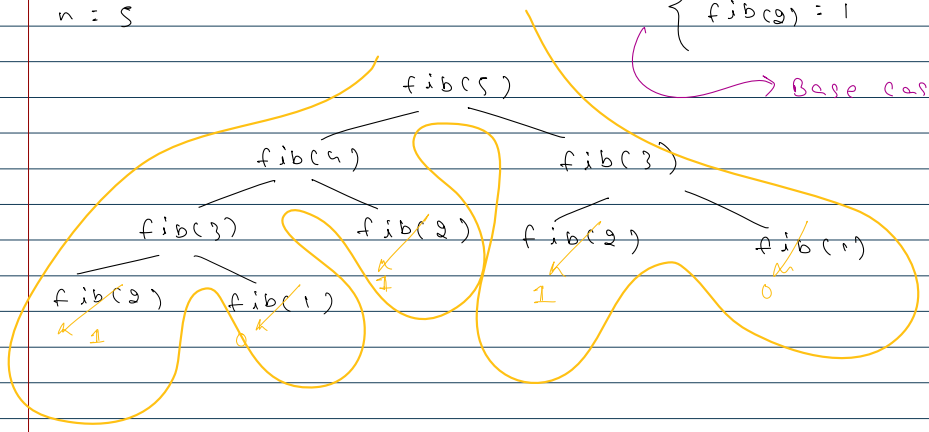
fib(87) = fib(86) + fib(85)
↓
87th term of the fibonacci series

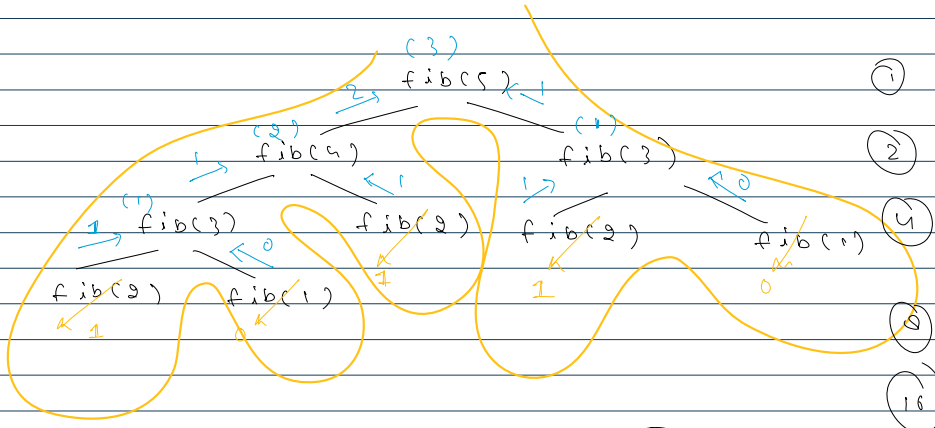
Recurrence relation → fib(n) = fib(n-1) + fib(n-2)

Recursive tree

n = 5

fib(1) = 0
fib(2) = 1
→ Base case/Condition





1 → 2^0
2 → 2^1
4 → 2^2
8 → 2^3
16 → 2^4

Time → $2^0 + 2^1 + 2^2 + 2^3 + 2^4 - - - 2^n$

```
public class Main {
    public static void main(String[] args) {
        int n = 5;
        System.out.println(fib(n));
    }

    public static int fib(int n){
        // if(n == 1){
        //     return 0;
        // }
        // if(n == 2){
        //     return 1;
        // }

        if(n==1 || n==2){
            return n-1;
        }

        return fib(n-1) + fib(n-2);
    }
}
```

Subsequence

Subarray ?

arr → [1, 2, 3] ⇒ [ ]
[1]
[2]
[3]
[1, 2]
[2, 3]
[1, 2, 3]

Subset ?

a b c → a, b, c | a b, b c | a b c | ~~a c~~

Subsequence ?

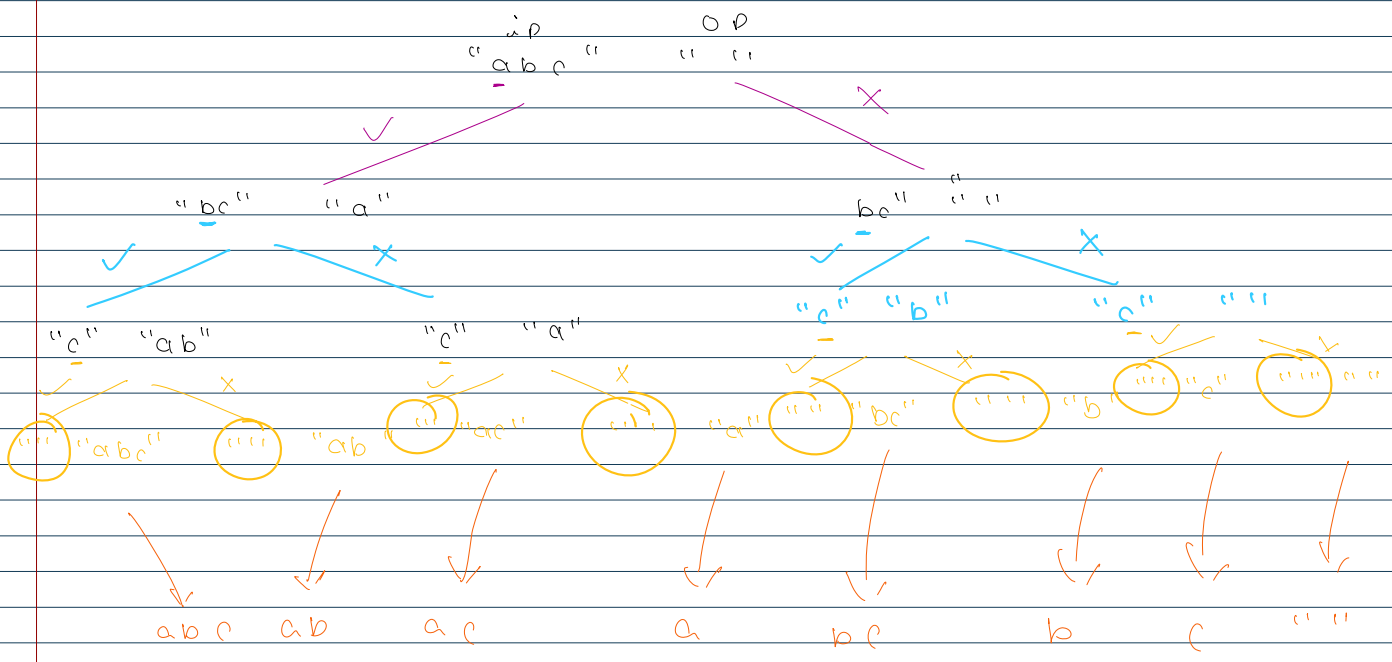
a b c → a, b, c | a b, b c, a c | a b c | ~~b a~~, ~~c b~~, ~~c a~~

Print all subsequence

Recursion → ?

abc ⇒ a → include this / don't include this → answer

choices ⇒ Recursion ⇒ decision ?



```java
public class Main {
    public static void main(String[] args) {
        String ip = "abc";
        String op = "";
        subsequence(ip, op);
    }

    public static void subsequence(String ip, String op){
        if(ip.length()==0){
            System.out.println(op);
            return;
        }
        char ch = ip.charAt(0);
        subsequence(ip.substring(1), op+ch); //Include
        subsequence(ip.substring(1), op) ; //Not include
    }
}
```

ip: bc
sub( )

ip: abc
op: ""
S(bc, "a")
S(bc, "")

ip: "abc"
op: ""
S(abc, "")
main

sub( )

abc => ip.substring(1)

||

bc

Print count of all subsequence

coin => ◯ => Head / Tail ~, flip => 2 possibilities

Head Tail

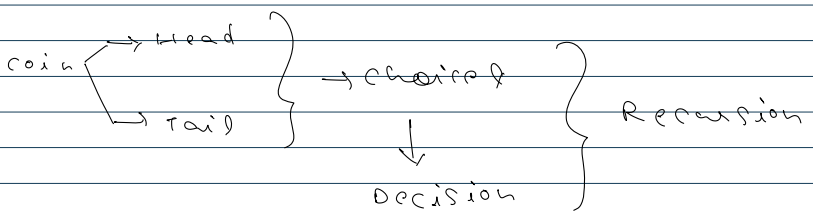
Print all possible outcomes of a coin flip

n = 2

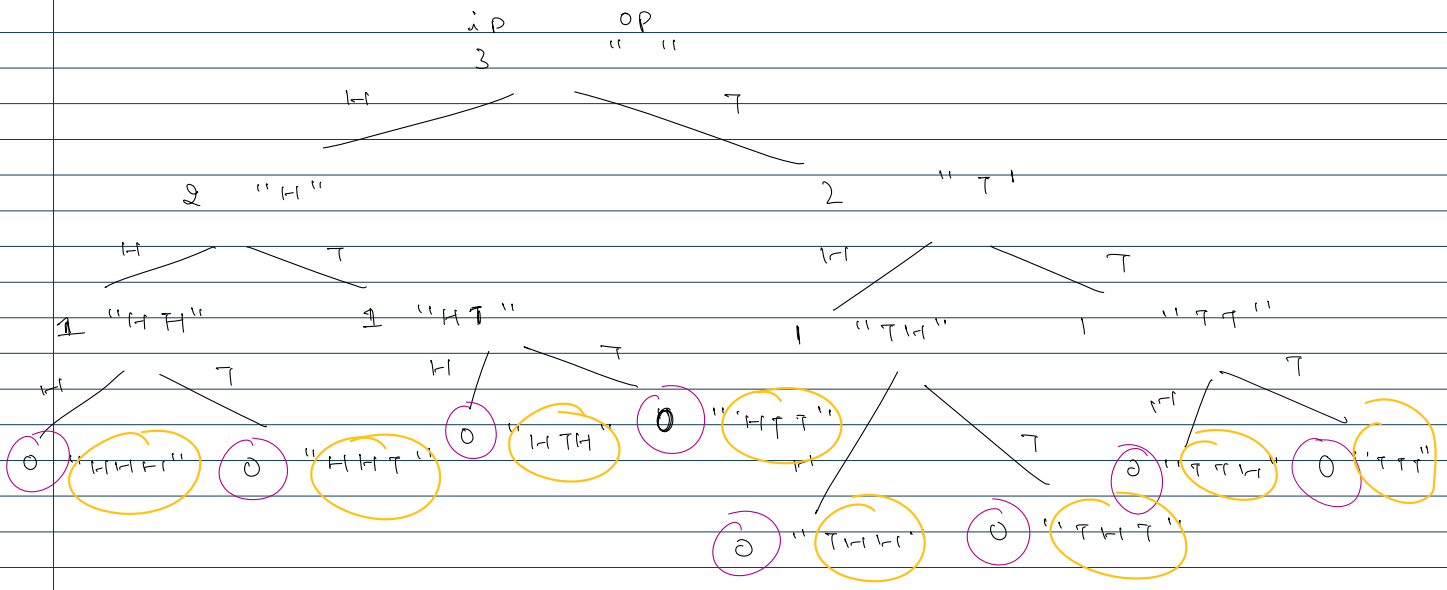
H H
H T
T H
T T

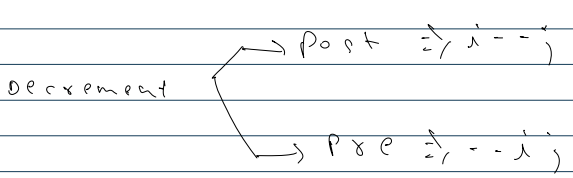
n = 3

H H H
H H T
H T H
H T T
T H H
T H T
T T H
T T T



Recursive Tree





784. Letter Case Permutation

Solved

Medium · Topics · Companies

Given a string `s`, you can transform every letter individually to be lowercase or uppercase to create another string.

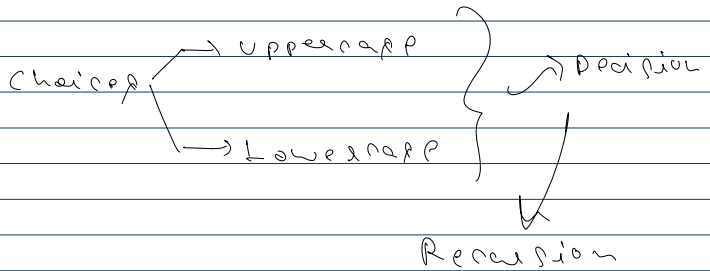
Return *a list of all possible strings we could create*. Return the output in **any order**.

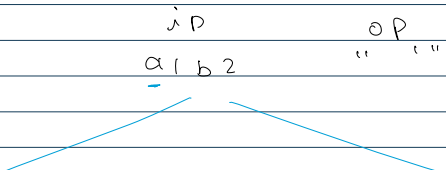
Example 1:

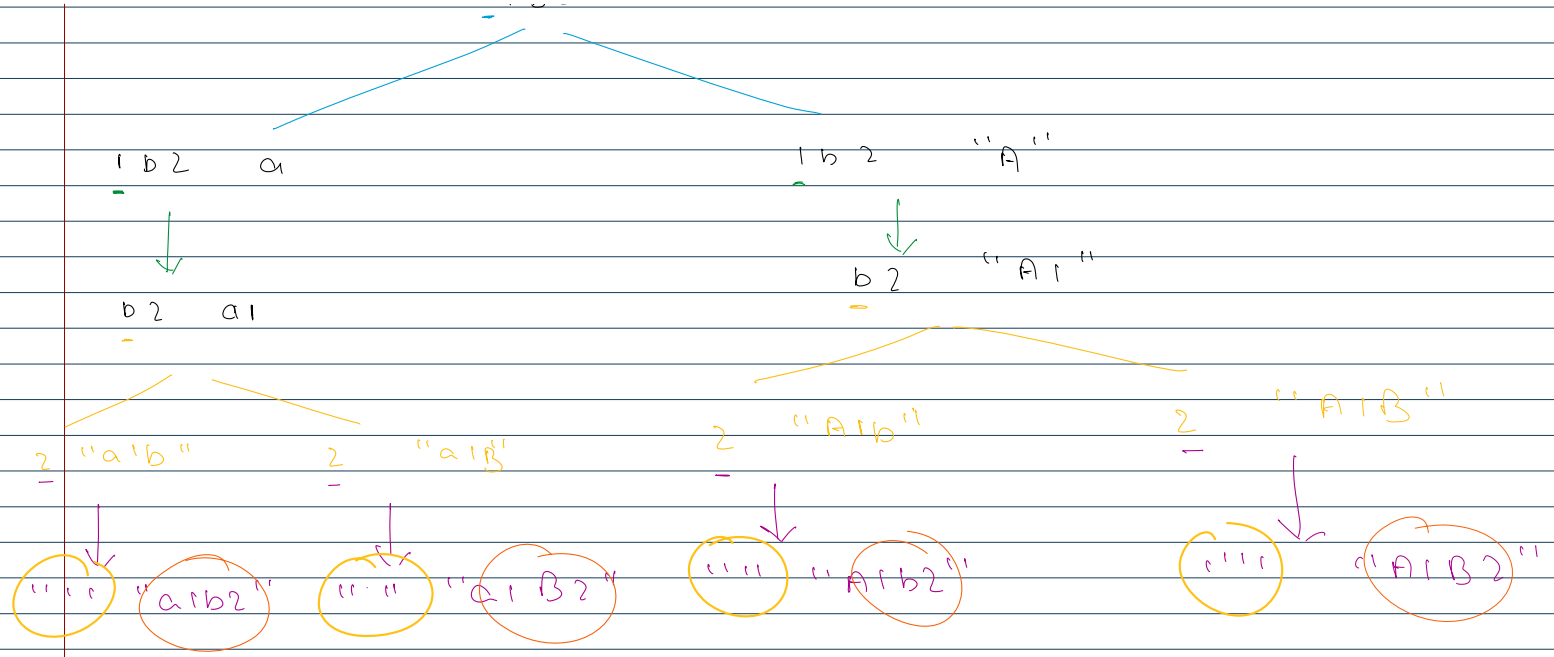
```
Input: s = "a1b2"
Output: ["a1b2","a1B2","A1b2","A1B2"]
```

Example 2:

```
Input: s = "3z4"
Output: ["3z4","3Z4"]
```






b 2 a
b 2 a 1
2 "a 1 b"
2 "a 1 B"
"" "a b 2"
"" "a 1 B 2"
b 2 "A"
b 2 "A 1"
2 "A 1 b"
"" "A 1 b 2"
2 "A 1 B"
"" "A 1 B 2"